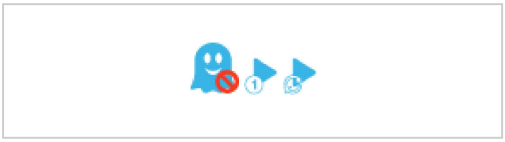
Search

Networking → Iperf
Last update: 10-12-2010

WWW.OPENMANIAK.COM
World Wide Made

If you like our tutorials, don't hesitate to support us and visit our sponsors!
Si vous aimez nos tutoriaux, n'hésitez pas à nous supporter et visiter nos sponsors!



Iperf is a tool to measure the bandwidth and the quality of a network link. Jperf can be associated with Iperf to provide a graphical frontend written in Java.

The network link is delimited by two hosts running Iperf.

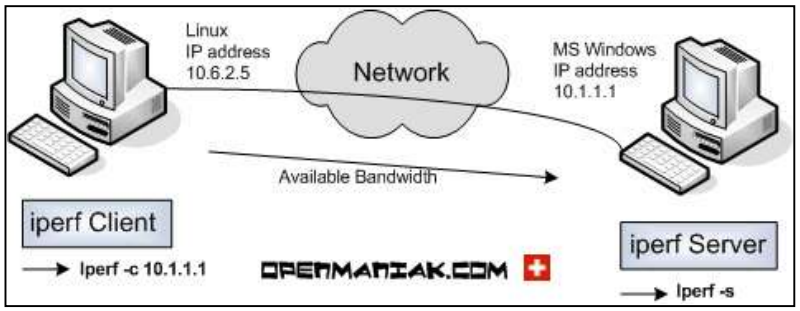
The quality of a link can be tested as follows:
- Latency (response time or RTT): can be measured with the Ping command.
- Jitter (latency variation): can be measured with an Iperf UDP test.
- Datagram loss: can be measured with an Iperf UDP test.

The bandwidth is measured through TCP tests.

To be clear, the difference between TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) is that TCP use processes to check that the packets are correctly sent to the receiver whereas with UDP the packets are sent without any checks but with the advantage of being quicker than TCP. Iperf uses the different capacities of TCP and UDP to provide statistics about network links.

Finally, Iperf can be installed very easily on any UNIX/Linux or Microsoft Windows system. One host must be set as client, the other one as server.

Here is a diagram where Iperf is installed on a Linux and Microsoft Windows machine.
Linux is used as the Iperf client and Windows as the Iperf server. Of course, it is also possible to use two Linux boxes.



Iperf tests:

no arg.	Default settings	-p, -t, -i	Port, timing and interval
-b	Data format	-u, -b	UDP tests, bandwidth settings
-r	Bi-directional bandwidth	-m	Maximum Segment Size display
-d	Simultaneous bi-directional bandwidth	-M	Maximum Segment Size settings
-w	TCP Window size	-P	Parallel tests
		-h	help

Jperf:

no arg.	Default settings
-d	Simultaneous bi-directional bandwidth
-u, -b	UDP tests, bandwidth settings

Default Iperf settings:
Also check "Jperf section.

By default, the Iperf client connects to the Iperf server on the TCP port 5001 and the bandwidth displayed by Iperf is the bandwidth from the client to the server.
If you want to use UDP tests, use the -u argument.
The -d and -r Iperf client arguments measure the bi-directional bandwidths. (See further on this tutorial)

→ Client side:

```
#iperf -c 10.1.1.1
```

```
------------------------------------------------------------
Client connecting to 10.1.1.1, TCP port 5001
TCP window size: 16384 Byte (default)
------------------------------------------------------------
[ 3] local 10.6.2.5 port 33453 connected with 10.1.1.1 port 5001
[ 3]  0.0-10.2 sec  1.26 MBytes  1.05 Mbits/sec
```

→ Server side:

```
#iperf -s
```

```
------------------------------------------------------------
```

→ **TOTAL**
Since dec 2006
1'942'871 Visitors
4'218'042 Pages

→ **Nov 2010 Stats**
82'909 Visitors
146'476 Pages
196 countries
Full statistics

→ Help us translate our tutorials!

→ JOIN the OpenManiak Team.

OM TEAM
→ **Director**:
Blaise Carrera
→ **Tutorials creation**:
Blaise Carrera
→ **Translaters:**
Giovanni Fredducci
Angel Chraniotis
Moham. H. Karvan
Alexandro Silva
Blaise Carrera
Andrei Chertolyas
Sergiy Uvarov
Nickola Kolev
Łukasz Nowatkowski
Ivo Raisr
Catalin Bivolaru
Bogdan A. Costea
Kirill Simonov
Oliver Mucafir
JaeYoung Jeon
Seungyoon Lee
Jie Yu & Si Cheng
Tao Wei
YukiAlex
Fumihito Yoshida
Muhammad Takdir
Çağdaş Tülek
→ **Auditors**
Leslie Luthi
Joe Anderson
Jennifer Ockwell
Nigel Titley
Alison Rees
Sabrina Barbey
→ **Webmaster:**
Blaise Carrera

```
Server listening on TCP port 5001
TCP window size: 8.00 KByte (default)
------------------------------------------------------------
[852] local 10.1.1.1 port 5001 connected with 10.6.2.5 port 33453
[ ID]  Interval       Transfer     Bandwidth
[852]  0.0-10.6 sec   1.26 MBytes  1.03 Mbits/sec
```

■ Data formatting: (-f argument)

The -f argument can display the results in the desired format: bits(b), bytes(B), kilobits(k), kilobytes(K), megabits(m), megabytes(M), gigabits(g) or gigabytes(G).
Generally the bandwidth measures are displayed in bits (or Kilobits, etc ...) and an amount of data is displayed in bytes (or Kilobytes, etc ...).
As a reminder, 1 byte is equal to 8 bits and, in the computer science world, 1 kilo is equal to 1024 (2^10).
For example: 100'000'000 bytes is not equal to 100 Mbytes but to 100'000'000/1024/1024 = 95.37 Mbytes.

➜ Client side:

#iperf -c 10.1.1.1 -f b

```
------------------------------------------------------------
Client connecting to 10.1.1.1, TCP port 5001
TCP window size: 16384 Byte (default)
------------------------------------------------------------
[ 3] local 10.6.2.5 port 54953 connected with 10.1.1.1 port 5001
[ 3]  0.0-10.2 sec  1359872 Bytes  1064272 bits/sec
```

➜ Server side:

#iperf -s

```
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 8.00 KByte (default)
------------------------------------------------------------
[852] local 10.1.1.1 port 5001 connected with 10.6.2.5 port 33453
[ ID]  Interval       Transfer     Bandwidth
[852]  0.0-10.6 sec   920 KBytes   711 Kbits/sec
```

Top of the page

■ Bi-directional bandwidth measurement: (-r argument)

The Iperf server connects back to the client allowing the bi-directional bandwidth measurement. By default, only the bandwidth from the client to the server is measured.
If you want to measure the bi-directional bandwidth simultaneously, use the -d keyword. (See next test.)

➜ Client side:

#iperf -c 10.1.1.1 -r

```
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
------------------------------------------------------------
Client connecting to 10.1.1.1, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[ 5] local 10.6.2.5 port 35726 connected with 10.1.1.1 port 5001
[ 5]  0.0-10.0 sec  1.12 MBytes  936 Kbits/sec
[ 4] local 10.6.2.5 port 5001 connected with 10.1.1.1 port 1640
[ 4]  0.0-10.1 sec  74.2 MBytes  61.7 Mbits/sec
```

➜ Server side:

#iperf -s

```
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 8.00 KByte (default)
------------------------------------------------------------
[852] local 10.1.1.1 port 5001 connected with 10.6.2.5 port 54355
[ ID]  Interval       Transfer     Bandwidth
[852]  0.0-10.1 sec   1.15 MBytes  956 Kbits/sec
------------------------------------------------------------
Client connecting to 10.6.2.5, TCP port 5001
TCP window size: 8.00 KByte (default)
------------------------------------------------------------
[824] local 10.1.1.1 port 1646 connected with 10.6.2.5 port 5001
[ ID]  Interval       Transfer     Bandwidth
[824]  0.0-10.0 sec   73.3 MBytes  61.4 Mbits/sec
```

Top of the page

■ Simultaneous bi-directional bandwidth measurement: (-d argument)
Also check the "Jperf" section.

To measure the bi-directional bandwidths simultaneousely, use the -d argument. If you want to test the bandwidths sequentially, use the -r argument (see previous test).
By default (ie: without the -r or -d arguments), only the bandwidth from the client to the server is measured.

➜ Client side:

#iperf -c 10.1.1.1 -d

```
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
------------------------------------------------------------
Client connecting to 10.1.1.1, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[ 5] local 10.6.2.5 port 60270 connected with 10.1.1.1 port 5001
[ 4] local 10.6.2.5 port 5001 connected with 10.1.1.1 port 2643
[ 4] 0.0-10.0 sec 76.3 MBytes 63.9 Mbits/sec
[ 5] 0.0-10.1 sec 1.55 MBytes 1.29 Mbits/sec
```

➜ Server side:

```
#iperf -s
```

```
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 8.00 KByte (default)
------------------------------------------------------------
[852] local 10.1.1.1 port 5001 connected with 10.6.2.5 port 60270
------------------------------------------------------------
Client connecting to 10.6.2.5, TCP port 5001
TCP window size: 8.00 KByte (default)
------------------------------------------------------------
[800] local 10.1.1.1 port 2643 connected with 10.6.2.5 port 5001
[ ID]  Interval       Transfer     Bandwidth
[800]  0.0-10.0 sec   76.3 MBytes  63.9 Mbits/sec
[852]  0.0-10.1 sec   1.55 MBytes  1.29 Mbits/sec
```

Top of the page

■ TCP Window size: (-w argument)

The TCP window size is the amount of data that can be buffered during a connection without a validation from the receiver.
It can be between 2 and 65,535 bytes.
On Linux systems, when specifying a TCP buffer size with the -w argument, the kernel allocates double as much as indicated.

➜ Client side:

```
#iperf -c 10.1.1.1 -w 2000
```

```
WARNING: TCP window size set to 2000 bytes. A small window size
will give poor performance. See the Iperf documentation.
------------------------------------------------------------
Client connecting to 10.1.1.1, TCP port 5001
TCP window size: 3.91 KByte (WARNING: requested 1.95 KByte)
------------------------------------------------------------
[ 3] local 10.6.2.5 port 51400 connected with 10.1.1.1 port 5001
[ 3]  0.0-10.1 sec   704 KBytes   572 Kbits/sec
```

➜ Server side:

```
#iperf -s -w 4000
```

```
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 3.91 KByte
------------------------------------------------------------
[852] local 10.1.1.1 port 5001 connected with 10.6.2.5 port 51400
[ ID]  Interval       Transfer     Bandwidth
[852]  0.0-10.1 sec   704 KBytes   570 Kbits/sec
```

Top of the page

▫

■ Communication port (-p), timing (-t) and interval (-i):

The Iperf server communication port can be changed with the -p argument. It must be configured on the client and the server with the same value, default is TCP port 5001.
The -t argument specifies the test duration time in seconds, default is 10 secs.
The -i argument indicates the interval in seconds between periodic bandwidth reports.

➜ Client side:

```
#iperf -c 10.1.1.1 -p 12000 -t 20 -i 2
```

```
------------------------------------------------------------
Client connecting to 10.1.1.1, TCP port 12000
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[ 3] local 10.6.2.5 port 58316 connected with 10.1.1.1 port 12000
[ 3]  0.0- 2.0 sec   224 KBytes   918 Kbits/sec
[ 3]  2.0- 4.0 sec   368 KBytes   1.51 Mbits/sec
[ 3]  4.0- 6.0 sec   704 KBytes   2.88 Mbits/sec
[ 3]  6.0- 8.0 sec   280 KBytes   1.15 Mbits/sec
[ 3]  8.0-10.0 sec   208 KBytes   852 Kbits/sec
[ 3] 10.0-12.0 sec   344 KBytes   1.41 Mbits/sec
[ 3] 12.0-14.0 sec   208 KBytes   852 Kbits/sec
[ 3] 14.0-16.0 sec   232 KBytes   950 Kbits/sec
[ 3] 16.0-18.0 sec   232 KBytes   950 Kbits/sec
[ 3] 18.0-20.0 sec   264 KBytes   1.08 Mbits/sec
[ 3]  0.0-20.1 sec   3.00 MBytes  1.25 Mbits/sec
```

➜ Server side:

```
#iperf -s -p 12000
```

```
------------------------------------------------------------
Server listening on TCP port 12000
TCP window size: 8.00 KByte (default)
------------------------------------------------------------
[852] local 10.1.1.1 port 12000 connected with 10.6.2.5 port 58316
[ ID] Interval Transfer Bandwidth
[852]  0.0-20.1 sec  3.00 MBytes  1.25 Mbits/sec
```

Top of the page

UDP tests: (-u), bandwidth settings (-b)
Also check the "Jperf" section.

The UDP tests with the -u argument will give invaluable information about the jitter and the packet loss.
If you don't specify the -u argument, Iperf uses TCP.
To keep a good link quality, the packet loss should not go over 1 %. A high packet loss rate will generate a lot of TCP segment retransmissions which will affect the bandwidth.

The jitter is basically the latency variation and does not depend on the latency. You can have high response times and a very low jitter. The jitter value is particularly important on network links supporting voice over IP (VoIP) because a high jitter can break a call.
The -b argument allows the allocation if the desired bandwidth.

➜ Client side:

```
#iperf -c 10.1.1.1 -u -b 10m
```

```
------------------------------------------------------------
Client connecting to 10.1.1.1, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size: 108 KByte (default)
------------------------------------------------------------
[ 3] local 10.6.2.5 port 32781 connected with 10.1.1.1 port 5001
[ 3]  0.0-10.0 sec  11.8 MBytes  9.89 Mbits/sec
[ 3] Sent 8409 datagrams
[ 3] Server Report:
[ 3]  0.0-10.0 sec  11.8 MBytes  9.86 Mbits/sec  2.617 ms  9/ 8409  (0.11%)
```

➜ Server side:

```
#iperf -s -u -i 1
```

```
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 8.00 KByte (default)
------------------------------------------------------------
[904] local 10.1.1.1 port 5001 connected with 10.6.2.5 port 32781
[ ID]  Interval       Transfer     Bandwidth       Jitter   Lost/Total Datagrams
[904]  0.0- 1.0 sec  1.17 MBytes  9.84 Mbits/sec  1.830 ms  0/ 837  (0%)
[904]  1.0- 2.0 sec  1.18 MBytes  9.94 Mbits/sec  1.846 ms  5/ 850  (0.59%)
[904]  2.0- 3.0 sec  1.19 MBytes  9.98 Mbits/sec  1.802 ms  2/ 851  (0.24%)
[904]  3.0- 4.0 sec  1.19 MBytes  10.0 Mbits/sec  1.830 ms  0/ 850  (0%)
[904]  4.0- 5.0 sec  1.19 MBytes  9.98 Mbits/sec  1.846 ms  1/ 850  (0.12%)
[904]  5.0- 6.0 sec  1.19 MBytes  10.0 Mbits/sec  1.806 ms  0/ 851  (0%)
[904]  6.0- 7.0 sec  1.06 MBytes  8.87 Mbits/sec  1.803 ms  1/ 755  (0.13%)
[904]  7.0- 8.0 sec  1.19 MBytes  10.0 Mbits/sec  1.831 ms  0/ 850  (0%)
[904]  8.0- 9.0 sec  1.19 MBytes  10.0 Mbits/sec  1.841 ms  0/ 850  (0%)
[904]  9.0-10.0 sec  1.19 MBytes  10.0 Mbits/sec  1.801 ms  0/ 851  (0%)
[904]  0.0-10.0 sec  11.8 MBytes  9.86 Mbits/sec  2.618 ms  9/ 8409  (0.11%)
```

Top of the page

Maximum Segment Size (-m argument) display:

The Maximum Segment Size (MSS) is the largest amount of data, in bytes, that a computer can support in a single, unfragmented TCP segment.
It can be calculated as follows:
MSS = MTU - TCP & IP headers
The TCP & IP headers are equal to 40 bytes.
The MTU or Maximum Transmission Unit is the greatest amount of data that can be transferred in a frame.
Here are some default MTU size for different network topology:
Ethernet - 1500 bytes: used in a LAN.
PPPoE - 1492 bytes: used on ADSL links.
Token Ring (16Mb/sec) - 17914 bytes: old technology developed by IBM.
Dial-up - 576 bytes

Generally, a higher MTU (and MSS) brings higher bandwidth efficiency

➜ Client side:

```
#iperf -c 10.1.1.1 -m
```

```
------------------------------------------------------------
Client connecting to 10.1.1.1, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[ 3] local 10.6.2.5 port 41532 connected with 10.1.1.1 port 5001
[ 3]  0.0-10.2 sec  1.27 MBytes  1.04 Mbits/sec
[ 3] MSS size 1448 bytes (MTU 1500 bytes, ethernet)
```

Here the MSS is not equal to 1500 - 40 but to 1500 - 40 - 12 (Timestamps option) = 1448

➜ Server side:

```
#iperf -s
```

Top of the page

■ Maximum Segment Size (-M argument) settings:

Use the -M argument to change the MSS. (See the previous test for more explanations about the MSS)

#iperf -c 10.1.1.1 -M 1300 -m

```
WARNING: attempt to set TCP maximum segment size to 1300, but got 536
------------------------------------------------------------
Client connecting to 10.1.1.1, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[ 3] local 10.6.2.5 port 41533 connected with 10.1.1.1 port 5001
[ 3]  0.0-10.1 sec  4.29 MBytes  3.58 Mbits/sec
[ 3] MSS size 1288 bytes (MTU 1328 bytes, unknown interface)
```

➔ Server side:

#iperf -s

Top of the page

■ Parallel tests (-P argument):

Use the -P argument to run parallel tests.

➔ Client side:

#iperf -c 10.1.1.1 -P 2

```
------------------------------------------------------------
Client connecting to 10.1.1.1, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[ 3] local 10.6.2.5 port 41534 connected with 10.1.1.1 port 5001
[ 4] local 10.6.2.5 port 41535 connected with 10.1.1.1 port 5001
[ 4]  0.0-10.1 sec  1.35 MBytes  1.12 Mbits/sec
[ 3]  0.0-10.1 sec  1.35 MBytes  1.12 Mbits/sec
[SUM] 0.0-10.1 sec  2.70 MBytes  2.24 Mbits/sec
```

➔ Server side:

#iperf -s

Top of the page

▫

■ Iperf help:

#iperf -h

```
Usage: iperf [-s|-c host] [options]
iperf [-h|--help] [-v|--version]

Client/Server:
-f  --format      [kmKM]  format to report: Kbits, Mbits, KBytes, MBytes
-i  --interval    #       seconds between periodic bandwidth reports
-l  --len         #[KM]   length of buffer to read or write (default 8 KB)
-m --print_mss            print TCP maximum segment size (MTU - TCP/IP header)
-p  --port        #       server port to listen on/connect to
-u  --udp                 use UDP rather than TCP
-w  --window      #[KM]   TCP window size (socket buffer size)
-B  --bind        "host"  bind to "host", an interface or multicast address
-C  --compatibility       for use with older versions does not sent extra msgs
-M  --mss         #       set TCP maximum segment size (MTU - 40 bytes)
-N  --nodelay             set TCP no delay, disabling Nagle's Algorithm
-V  --IPv6Version         Set the domain to IPv6

Server specific:
-s  --server              run in server mode
-U --single_udp           run in single threaded UDP mode
-D --daemon               run the server as a daemon

Client specific:
-b --bandwidth    #[KM]   for UDP, bandwidth to send at in bits/sec (default 1 Mbit/sec, implies -u)
-c --client       "host"  run in client mode, connecting to "host"
-d --dualtest             Do a bidirectional test simultaneously
-n --num          #[KM]   number of bytes to transmit (instead of -t)
-r --tradeoff             Do a bidirectional test individually
-t  --time        #       time in seconds to transmit for (default 10 secs)
-F --fileinput    "name"  input the data to be transmitted from a file
-I  --stdin               input the data to be transmitted from stdin
-L --listenport   #       port to recieve bidirectional tests back on
-P --parallel     #       number of parallel client threads to run
-T --ttl          #       time-to-live, for multicast (default 1)

Miscellaneous:
-h --help                 print this message and quit
-v --version              print version information and quit
```

Top of the page

JPERF

Jperf is a graphical frontend for Iperf written in Java.

■ 1. Installation:

Download Jperf.

➜ Linux

Uncompress the downloaded file:

```
#tar -xvf jperf2.0.0.zip
```

Launch Jperf.

```
#cd jperf2.0.0
#./jperf.sh
```

If you have the following message, it means that you need to install Iperf with: "apt-get install iperf"

Iperf is probably not in your Path!
Please download it here 'http://dast.nlanr.net/Projects/Iperf/'
and put the executable in your PATH environment variable.

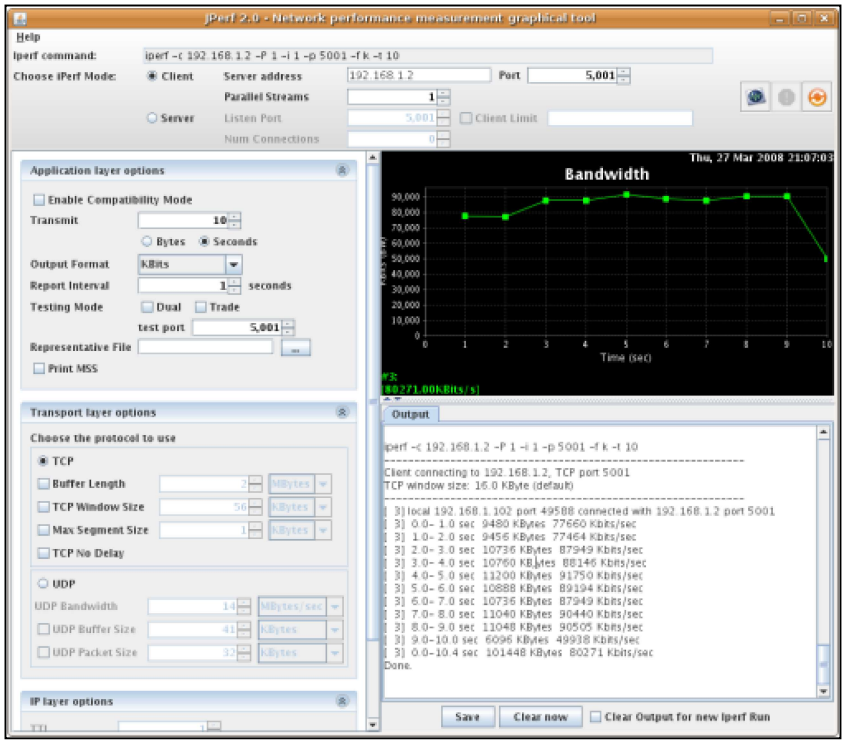
➜ Microsoft Windows

Uncompress the downloaded file with your favorite program.
Access the uncompressed folder called by default "jperf2.0.0" and double-click on "jperf.bat".
Note that the iperf utility is already present in the /bin folder.

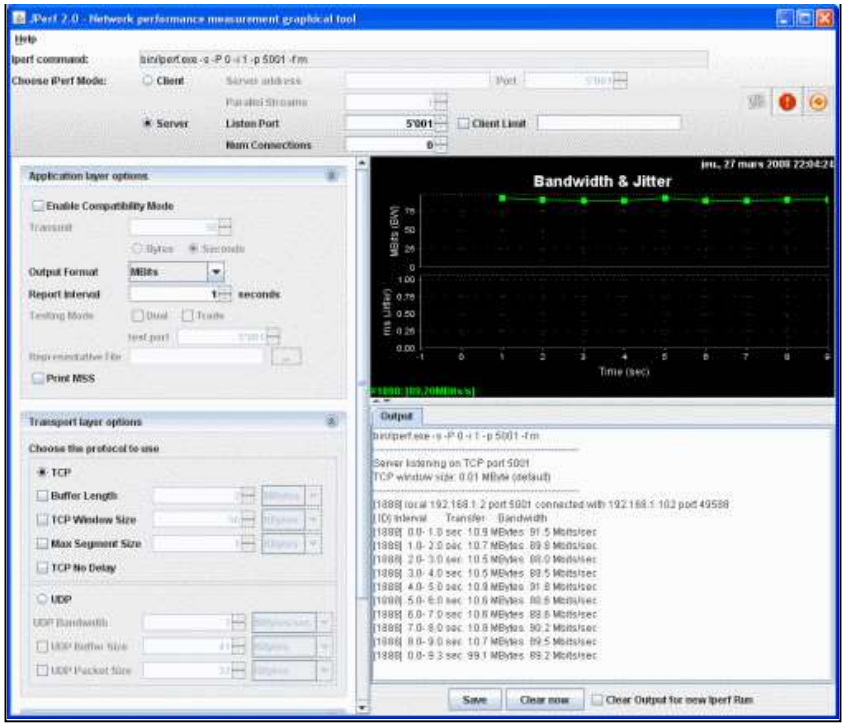
■ 2. Examples:

➜ Default settings, bandwidth measurement:
Also check "Iperf" section for more details.

- Linux client:



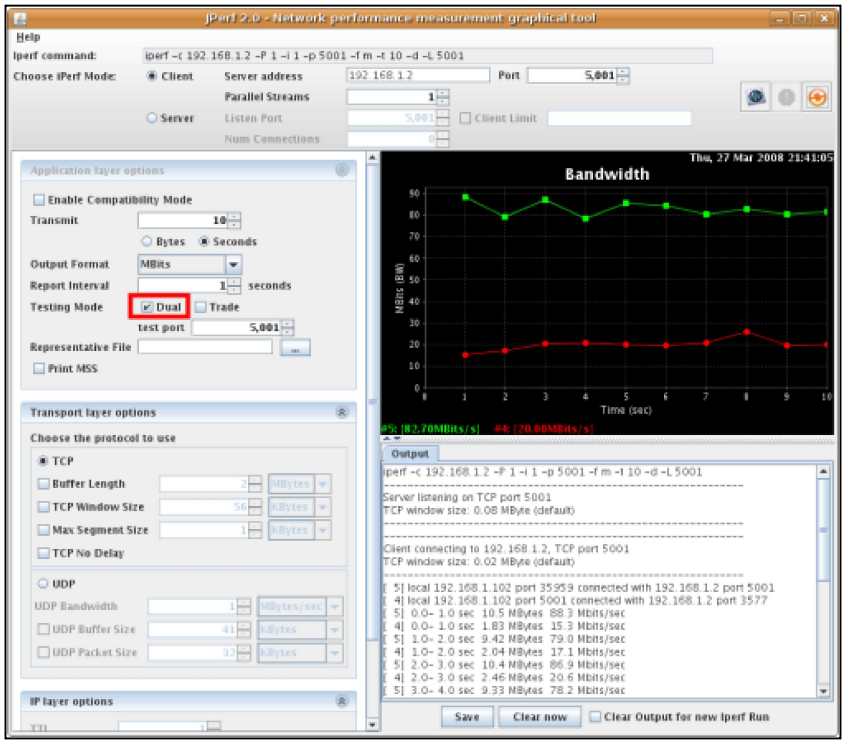
- Windows server:



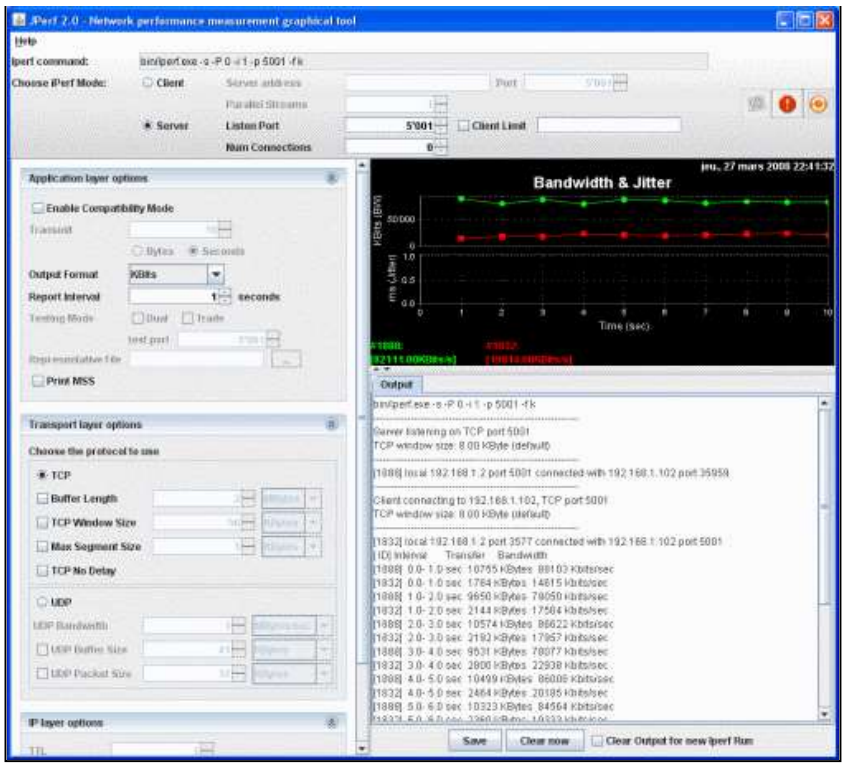
Top of the page Jperf

➜ Simultaneous bi-directional bandwidth measurement:
Also check "Iperf" section for more details.

- Linux client:



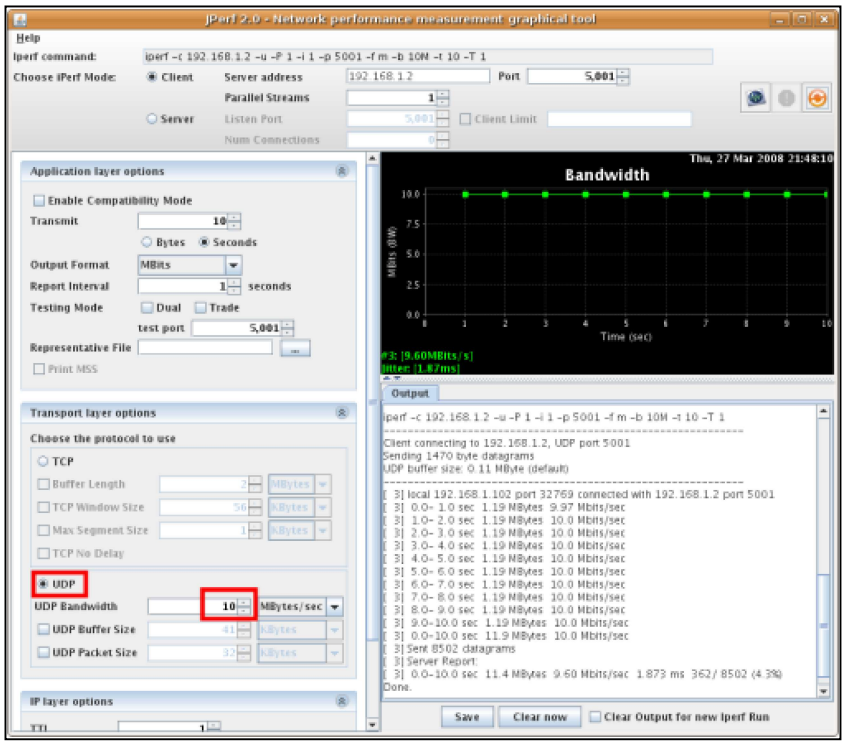
- Windows server:



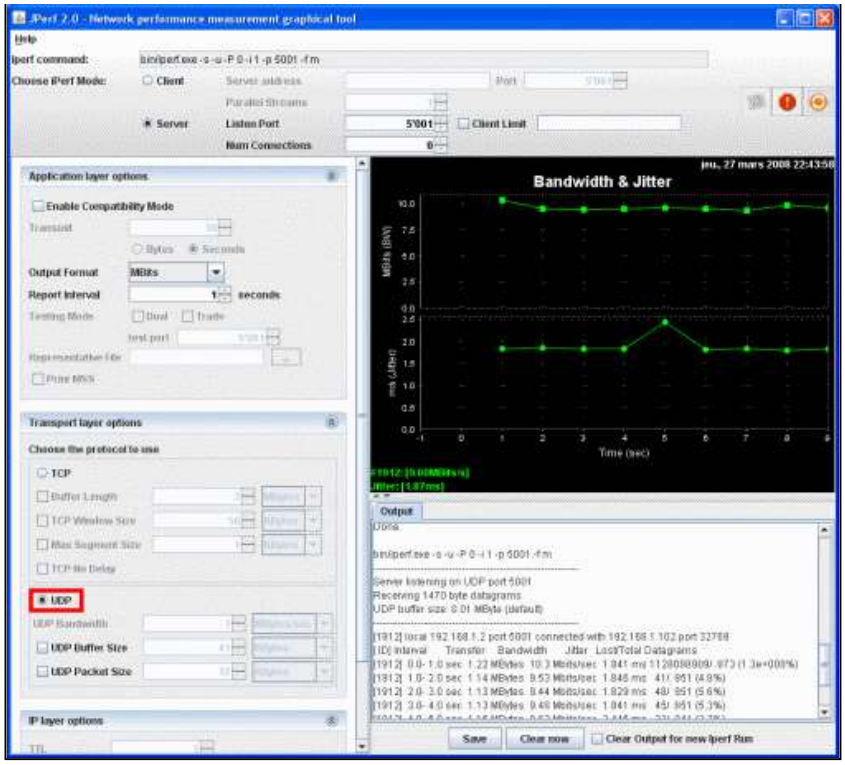
Top of the page Jperf

Jitter measurement:
Also check "Iperf" section for more details.

- Linux client:



- Windows server:



Top of the page Jperf

If you liked our tutorials, don't hesitate to support us and visit our sponsors!
Si vous aimez nos tutoriaux, n'hésitez pas à nous supporter et visiter nos sponsors!